CSE120 Wi19 Midterm Review Solutions

1. Lightbot questions

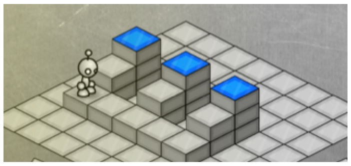
   a. How does the Lightbot keep track of where it is in the program?

      The program counter (in Lightbot, the yellow box)

   b. Lightbot (and computers) can do only a small amount of instructions; how do programmers avoid the tedium of using these instructions all the time?

      We break up groups of instructions into units called functions, so that instead of writing every instruction each time, we can just call the function containing them.

   c. If F_1 is a Lightbot function that contains: `Left, Jump, Jump, Left`. Write an F_2 so the solution for the problem at right is `3:(`F_2`).`



      F2: 2:(F1), 2:(Step)

2. Functional abstraction is recognizing a concept/subtask in a sequence of operations and giving it a name; why do we bother to abstract?

   - reduce complexity
   - give intuitive names to subproblems
   - solve the whole problem by composing functions

3. Give two ways in which the functions of Processing are "more powerful" than the functions of Lightbot – what can a Processing function do that a Lightbot function can't?

   - take arguments
   - return values

4. What is the difference between an *active* and a *static* program in Processing? Provide some examples in your answer.

   An active program contains **`setup`**`()` and **`draw`**`()` functions and runs continually, whereas a static program executes once and then is done. This allows us to make changes between frames. For instance, a program where clicking the mouse changes the color or a program that shows a character moving across the canvas are active programs.

5. Is the following line of code correct, incorrect, or unknown? Explain your choice.

```
line(150, 150, mousex, mousey);
```

   Incorrect – `mouse`**`X`** and `mouse`**`Y`** need to be capitalized.

6. `fill()` can take up to 4 arguments. In order, what do these arguments set/determine?

   red value, green value, blue value, and opacity (all from 0-255)

7. Describe the following background colors

   a. `background(255, 0, 0);` Bright red
   b. `background(64);` Dark grey
   c. `background(0, 0, 64);` Dark blue

8. Define the term *variable*.

   Like a box, a variable is a named storage space that holds a value of its declared type.

9. State the data type of each of the values below:

   a. `3.14` `float`
   b. `color(0)` `color`
   c. `-3` `int`
   d. `'x'` `char`
   e. `true` `boolean`

10. State the data type of each of the values returned by the system variables below:

    a. `mouseX` `int`
    b. `width` `int`
    c. `key` `char`
    d. `keyPressed` `boolean`
    e. `frameCount` `int`

11. In your own words, describe what the statement below does.

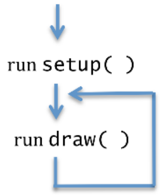
```
int ra = 200;
```

    Create (declare) a new variable of type `int` with name `ra` and give it (initialize it to) the value `200`.

12. What are the possible values that a `boolean` variable can hold?

    `true` or `false`

13. Explain the diagram below in your own words.



In Processing, the **`setup`**`()` function is run once when you first run your program, then the **`draw`**`()` function is run continuously in an infinite loop.

14. Briefly explain what the following do or what they are used for:

a. `%` mod: returns the remainder of a division, i.e. 5 % 3 = 2
b. `min()` returns the smallest of two (or more) arguments
c. `||` or: evaluates to true if either the left or right side is true; otherwise false
d. `;` semicolon: terminates a statement/command in the program
e. `{` and `}` curly braces: used to mark the beginning and end of code "blocks", e.g. for if-statements, for-loops, or function bodies.

15. How many parameters does the function header below have?

```
void ninja(int x_pos, int y_pos, color goggles)
```

3 parameters: `x_pos`, `y_pos`, and `goggles`.

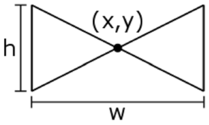
16. Write a piece of code that will display a circle 20 pixels wide that flashes random colors.

```
void draw() {
   fill(random(255), random(255), random(255));
   ellipse(100, 100, 20, 20);
}
// background() is optional since circle always drawn in same place
```

17. If I wanted to write a program that draws 4 identical faces on the screen, explain in words (not code) the best way to go about this. What kind of programming constructs should you use?

Create a *function* that contains the drawing commands for the face. Add *parameters* `x_pos` and `y_pos` to the function and update the drawing commands to use these parameters/variables. In **`draw`**`()`, call your face-drawing function four consecutive times with different *argument* values.

18. Complete the program below that draws the bowtie shown at right:



```
float x = 20, y = 20, w = 40, h = 20;

triangle( x, y, x-w/2, y+h/2, x-w/2, y-h/2 );

triangle( x, y, x+w/2, y+h/2, x+w/2, y-h/2 );
```

19. In the code below, which variables are *parameters*? Which variables are used in *arguments*?

```
color gray (int shade, int number) {
  record = max(number, record);
  return color(shade+2, shade+2, shade+2);
}
```

`shade` and `number` are parameters of `gray()`. `number` and `record` are passed as arguments to `max()` and `shade` is passed as an argument to `color()`.

20. Do the two statements below mean the same thing? Explain your answer.

```
seconds = seconds + 1;

seconds + 1 = seconds;
```

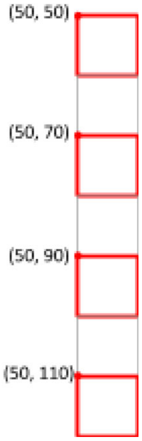
No, the second statement is not valid – you cannot assign a value to something that isn't a variable.

21. Given the for-loop below, draw the result. Label at least one point per shape.

```
for (int i = 0; i < 4; i = i + 1) {
  rect(50, 50+20*i, 10, 10);
}
```

This code will draw 4 10x10 squares with 10 pixels of vertical distance between each one:

```
rect(50, 50, 10, 10);
rect(50, 70, 10, 10);
rect(50, 90, 10, 10);
rect(50, 110, 10, 10);
```



22. If the 4 in the code above (Question 21) is changed to a 5, what will change in the drawing?

One more square will be drawn below the others at position (50,130).

23. Write a snippet of code equivalent to Question 21 that uses a while-loop instead of a for-loop.

```
int i = 0;
while (i < 4) {
   rect(50, 50+20*i, 10, 10);
   i = i + 1;
}
```

24. If `void dog(float xpos, float ypos)` is a function that draws a dog at `(xpos, ypos)`, write a for-loop to draw ten dogs in a row, with a dog every 100 pixels in the *horizontal* direction.

```
for (int i = 0; i < 10; i++) {
   dog(i * 100, 100);  // (0, 100) is the starting position
}
```

25. We're writing a function that computes the *factorial* of an integer. The factorial of a number n is written as $n!$ and is defined as $n! = n \times (n-1) \times \ldots \times 1$. For example, $4! = 4 \times 3 \times 2 \times 1 = 24$. Also note that $0! = 1$.

    Some code is already provided for you. Your goal is to finish filling in the function:

```
int factorial(int n) {
  int product = 1;
  for (int i = n; i > 0; i = i - 1) {
    product = product * i;
  }
  return product;
}
```

26. Name one benefit of digitizing information.

    Easy to find/search files, easy to copy, doesn't degrade easily, doesn't use physical space, etc.

27. What is an Internet packet?

    A small, fixed-size unit of information sent over the internet. It generally contains the destination IP address, packet number, and data being sent. Large pieces of information are broken into multiple packets and reassembled at the destination.

28. What is a "byte"?

    A byte is 8 bits (binary digits) of data. It can represent up to 2^8 = 256 things (*e.g.* the numbers 0-255).

29. The binary number `10 0101` represents what decimal number?

    $1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 32 + 4 + 1 = 37$.

30. What are all the possible three-bit binary sequences?

    `0b000`, `0b001`, `0b010`, `0b011`, `0b100`, `0b101`, `0b110`, and `0b111`

31. Name or describe a famous algorithm that you use on a regular basis.

    RSA encryption, Google's search algorithm (PageRank), Facebook's News Feed algorithm, etc.

32. Give an example of a digital rights management (DRM) technology. Briefly explain how it is intended to uphold copyright.

    Software serial keys: can't use the software until you get a key from the developer.
    Persistent online authentication: online server can check your identity (*e.g.* your IP address).
    Etc.